Usage Behavior of a Large-Scale Scientific Archive

Ian F. Adams*, Brian A. Madden*, Joel C. Frank*, Mark W. Storer†, Ethan L. Miller*, Gene Harano‡
*University of California, Santa Cruz †NetApp ‡NCAR

***Abstract*—Archival storage systems for scientific data have been growing in both size and relevance over the past two decades, yet researchers and system designers alike must rely on limited and obsolete knowledge to guide archival management and design. To address this issue, we analyzed three years of file-level activities from the NCAR mass storage system, providing valuable insight into a large-scale scientific archive with over 1600 users, tens of millions of files, and petabytes of data.**

Our examination of system usage showed that, while a subset of users were responsible for most of the activity, this activity was widely distributed at the file level. We also show that the physical grouping of files and directories on media can improve archival storage system performance. Based on our observations, we provide suggestions and guidance for both future scientific archival system designs as well as improved tracing of archival activity.

I. Introduction

Over the past two decades, scientific archives have grown from tens of terabytes to 10–100 petabytes, and are rapidly approaching exabyte-scale. However, our understanding of their behavior is woefully out-of-date. Since 1993, there have been only two studies that have looked explicitly at day-to-day activities on a large-scale scientific archive, each limited in their application to modern scientific archive design. Adams *et al.* were stymied by the coarse granularity of their data, preventing them from examining user and file level behaviors [1]. Frank *et al.* focused on evolutionary trends over time rather than a detailed analysis of a modern system's behavior [2]. Because of this lack of knowledge, designers of modern scientific archives are forced to rely on unverified assumptions and potentially obsolete workload studies from decades ago.

To address this issue, we examine the usage behavior of the National Center for Atmospheric Research's (NCAR) mass storage system (MSS). With over 1600 users and petabytes of data stored across millions of files, access traces from the NCAR MSS provide an excellent opportunity to understand the characteristics of a large-scale scientific archive. Using three years of activity logs running from 2008 through 2010 and extensive communication with administrators of the NCAR MSS, we have completed the first detailed study of a scientific archive, including user and file-level behaviors, in nearly 20 years.

At the system-wide level, we note that over 50% of activity was automated in nature, consisting of migrations of data to and from a disk cache and between tapes. Interestingly, while migrations are based upon administrative policy and cache availability, they are ultimately driven by user activities. They offer the best of both worlds as an easy target for optimization because they are both predictable and latency insensitive. Our analysis also shows that the notion of "Write-Once, Read-Maybe" is misleading in a modern scientific archive: at the system level hardware migrations inevitably cause data to be read and written *en masse*.

At the user level, we find that a subset of users and their associated sessions were responsible for the vast majority of activity in the system, and that user activities tend to occur in groups at the same directory depth. We show that the typical user-session, however, was rather modest in both the number of activities and the volume of data accessed, suggesting two improvements for systems based on offline or powered-down media. First, physical grouping based upon user and directory depth may be useful for significantly improving the performance of an archive based on offline media such as spun down disk or tape. Second, it may be helpful to have a batch style interface to aid in grouping and scheduling accesses to the archive.

At the file level, we discovered that a nontrivial fraction (5%) of files receive updates or overwrites of their data. We also found that around 15% of files created during the trace were deleted, typically within a year of creation. While this does not completely discount the notion of "Write-Once" in an archive, this shows that it is not unequivocally true either. We also show that activities were nearly uniformly distributed among files, with only a small fraction of files being highly active within the system. This contrasts strongly with typical enterprise and personal storage system workloads in which a small fraction of files are responsible for the majority of activity. This suggests that naïve read caching would be largely ineffective. In fact, the MSS architecture is organized to use the disk cache primarily as a write-buffer rather than a read cache.

The rest of the paper is structured as follows. In Section II we discuss prior workload studies and how they relate to our work. Section III provides an overview of the datasets we use in our analysis, and the system they came from. We present our observations and microanalysis in Section IV. In Section V we draw higher-level conclusions as well as discuss some lessons learned during our analysis. Section VI covers our future work and in Section VII we conclude.

SC12, November 10-16, 2012, Salt Lake City, Utah, USA
978-1-4673-0806-9/12/ $31.00 ©2012 IEEE

II. Related Work

While there have been few long-term studies of modern scientific archival storage systems, there have been long-term studies completed on other types of systems. Agrawal, *et al.* examined 5 years of snapshots from personal computing machines [3], and Gibson, *et al.* looked at long-term patterns on general-purpose UNIX machines [4]. These traces were gathered on file systems used for desktop computing; equally important, they are studies of *actively used* data, not archival data. While they provide a good view of how desktop users access data over a long period, they are not applicable to archival scientific data. Cho, *et al.* observed file system activities over the course of a year on a large scientific cluster [5]. While this study was in an HPC environment, it focused on relatively short-term home directory and scratch file systems with a total capacity under 100 TB. Again, these findings cannot be extended to scientific archival storage, with a capacity 2–3 orders of magnitude larger and much longer-term retention.

In addition to the long-term studies mentioned above, there have also been a variety of shorter-term workload studies in the last decade, ranging from a single day to a few months. Dayal [6] did an analysis of metadata snapshots from several scientific file systems including archives which, while providing a good one-time view, was unable to shed light on access patterns over time. Wang, *et al.* [7] explored short-term file system usage behavior of HPC applications, but did not examine aggregate usage of the file system by all users, and did not explore long-term behavior. Anderson [8] studied high-performance clusters used for animation rendering, which we cannot generalize to archival storage behaviors. As with Anderson's work, studies by Leung, *et al.* [9] and Chen, *et al.* [10] were relatively short term and on enterprise storage systems (in fact, both were based on the same file system trace), limiting their applicability to scientific archives.

Prior published studies that are both long-term and focused on scientific archives are either out of date, between 20 and 30 years old, or of limited applicability. In the early 1980s, Smith looked at long-term file reference patterns at the Stanford Linear Accelerator [11]. Miller and Katz examined file migration within the NCAR MSS in 1993 [12] and Jensen and Reed did a similar study of a system at the National Center for Supercomputing Applications [13], also in 1993. While these older studies are useful for comparison and evolutionary analyses [2], the small scale of these systems limits their relevance to modern archive design. For example, the 1993 NCAR system contained less than 30 TB of data, while the disk cache *alone* in the 2010 system can store nearly 100 TB.

More recently, there have been two studies that have examined scientific archive behavior. Adams, *et al.* looked at the modification behavior of an archive at LANL, but, due to the coarse granularity of the data they could not analyze per-user or file behaviors [1]. Frank, *et al.* looked at high-level behaviors and some file-level activities using the same dataset as this work, but focused on repeating Miller and Katz's 1993 analysis of the NCAR MSS to examine evolutionary trends [2, 12]. Our work differs from Frank, *et al.* in that it focuses solely on the 2010 NCAR data, and explores the impact of automated behaviors such as file migration as well providing detailed analyses of user and file level behaviors.

III. Background

We first define a consistent set of terminology and definitions that we use throughout this paper, adapted from Adams *et al.* [1]. Our dataset is comprised of daily logs of *actions*, each of which is an operation from a single user on exactly one file, *e.g.* the creation of a file. Users may be humans or purely automated processes. All the files and data stored are referred to as the *corpus*. The combination of hardware and software that store the corpus is the *archive*. We refer to the aggregate knowledge about the archive as a *sketch*. A sketch includes the logs as well as out-of-band information obtained through communication with system architects and administrators.

A. NCAR MSS Overview

The National Center for Atmospheric Research (NCAR) is dedicated to meteorological and climate research and its associated impacts. Our analysis is focused on the mass storage system (MSS), a tape-based storage archive used for storing a variety of datasets for months to years.

The MSS consists of tape libraries with a disk cache "in front" to serve as a write and read cache for files under 10 GB and is illustrated in Figure 1. For files under 10 GB, clients write directly to the disk cache; the files are later migrated to tape via an automated migration process. Files over 10 GB are written directly to tape. For reads, the first read of a file goes directly from tape to the user. If another read of the *same* file occurs within 24 hours, it is then cached on disk, assuming its size is under 10 GB. Because of these policies, the cache is primarily utilized as a write-buffer, which in turn generates large amounts of automated migration traffic as data is moved to tape.

Users interact with the MSS through a simple queue that addresses user requests in a FIFO manner. Files are written to any currently mounted media that has sufficient space. Reads are queued to a specific tape or disk volume, allowing for multiple reads from a single tape mount. Given the FIFO nature of request handling we see a relatively "pure" ordering of user-requests. Thus, our analysis reflects user behaviors with less noise than an analysis of a system that aggressively reorders and groups requests.

As an example of MSS operation, consider a user wanting to archive a 1 GB file `myData.dat`. Initially, `myData.dat` is created on the disk cache. As space is needed on the disk cache, the `myData.dat` file will be copied to the first tape that has sufficient space and subsequently removed from the disk cache. If `myData.dat` is later read, the initial read will go directly from tape to the user, bypassing the cache. If the file is read a second time within 24 hours, `myData.dat` will be cached on disk.

Purges (permanent deletions) of files from the system are based upon a file's retention policy, or done by explicit request

Date	Hardware
Jan 2008	5 StorageTex Powderhorn Silos 40 TB Disk Cache 70 STK 9940B Drives
Jun 2008	5 StorageTex Powderhorn Silos 100 TB Disk Cache 70 STK 9940B Drives
Jan 2009	5 StorageTex Powderhorn Silos † 2 SL8500 Tape Libraries 100 TB Disk Cache 70 STK 9940B Drives † 70 STK T10000B drives
Mar 2010	2 SL8500 Tape Libraries 100 TB Disk Cache 70 STK T10000B drives

TABLE I: Evolution of MSS hardware. January 2008 is the start state of the system. With the addition of the SL8500 libraries, the corpus was migrated from the Powderhorn libraries in preparation for their decommissioning. † denotes hardware in the process of decommissioning.

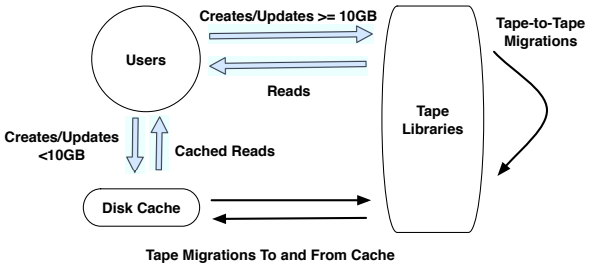


Fig. 1: This figure provides an overview of the NCAR MSS. Note that all files under 10 GB that are created or written to are cached first on the disk cache, while reads are only cached if a file is read twice within 24 hours. Files greater than 10 GB in size are written directly to tape, and all reads initially come from tape if they are not already present on the disk cache.

from a user. Each file has a retention policy describing the number of days it should be retained in the archive, *e. g.*, 360 days. If a file passes its retention period without extension, it is considered to be in the *trash*. After 30 days in the trash, the file is permanently deleted from the system and its removal is logged as an action. Note that the default retention period and trash time are system parameters set by NCAR staff, though actual retention periods can be modified by users.

The hardware evolved significantly during the logged period, as summarized in Table I. As we discuss further in Section IV-A, the data migrations associated with this technology change are not noted in our logs. No further hardware changes occurred from March 2010 until the end of our dataset.

The data corpus stored on the MSS is comprised of approximately 80% simulation output, 15% observational data for validation and seeding of simulations, and 5% system backups. At the beginning of our dataset in January 2008, the corpus was known to hold approximately 4 PB (petabytes) of data; at the end of our dataset, it held approximately 11.7 PB of data in 69 million files. However, approximately 3.3 PB were duplicate bytes: extra copies of files for reliability purposes. A typical long-term use case of data stored in the archive is storing simulation output and retrieving it 2–5 years later for re-analysis and validation.

B. Methodology Overview

For the purpose of our analysis we group actions into three broad categories. The first, which we call *user activities*, consist of actions that create, read or update a file's data. These actions almost always come from end-users of the archive. The second is *migration activities*: automated moves of data to and from the disk cache, as well as a small number of tape-to-tape migrations and integrity checks of data stored on tape. The third category of actions are *purges*, the permanent deletion of files within the archive. We first discuss the actions associated with user and migration activities. We do not go into greater detail on purges since there is only a single action, also called a purge, associated with that category.

There are three actions that we group under the category of user activities with which we are concerned: creates, reads, and writes. A create is the ingestion of a file and all of its associated data into the MSS. A write is an *update* to a file already in existence in the archive. When we refer to user activities, we use the terms write and update interchangeably. A read is simply a read of a file's data.

There are two types of actions in the category of migration activities. The first is a *migration read*, which is simply a read of a file in preparation for writing the file elsewhere in the system. The second is a *migration write*, which is a write of the file data read from a preceding migration read. With the exception of a small fraction of aborted or mis-logged activities, every migration write has a corresponding migration read. Thus, most user activities (those that are to the disk cache) have at *least* two associated migration actions that will follow it at some point. As we discuss further in Section IV-A, these migration activities are actually responsible for most of the data movement in the system.

Throughout our workload study when we calculate the number of bytes involved in any action or group of actions, we are summing the file sizes. As we lack data describing how much data is actually moved or written, we use the file size to approximate an upper bound.

C. Workload Overview

From January 1, 2008 to December 31, 2010 we identified approximately 50.5 million unique files in our logs. From our sketch, we know the total corpus contained approximately 69 million files at the end of 2010; however our logs only account for files acted upon during the 3 years of observation. We only know of the total corpus file count from out-of-band communication with administrators. As we discuss further in Section V, these communications were invaluable in aiding our understanding and observations.

Figure 2 provides a high-level overview of what the workload looks like over the course of a week. We see strong diurnal patterns linked to the workday [2], and note that there is significantly more activity due to file migrations than user actions, which we discuss further in Section IV.

Activity Type	Count	% of Total
All Actions	189,364,952	100%
User Acts. Total	65,980,770	34%
User Reads	22,272,374	12%
User Writes (Updates)	5,797,550	3%
User Creates	37,910,846	20%
Migrate Acts. Total	111,895,464	59%
Migrate Read	55,952,916	30%
Migrate Writes	55,942,548	29%
Purges Total	11,488,718	6%

TABLE II: Summary of actions observed during the trace period. Percentage of total is the fraction of all actions counted. Migrate and user totals are the total number of actions within those categories. Subtotal percentages are approximate to ensure a sum of 100%.

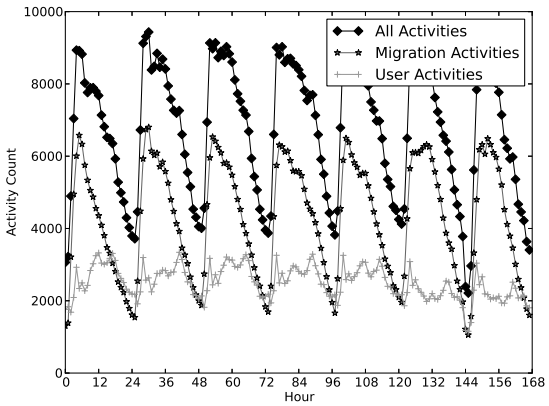


Fig. 2: Averaged hourly activity rates for actions, exclusive of purges. The "all activities" line is the sum of the user and migration activities for that hour. Hour 0 corresponds to Monday at midnight.

Figure 3 is a cumulative distribution function (CDF) of file sizes observed over the 3 years of data. *Count* refers to the fraction of files of a given size, while *volume* refers to the amount of space taken up by files of a given size [2]. We found that 80% of observed files are smaller than 100 MB, but most space is taken by files larger than 100 MB. The typical file size in the NCAR MSS is larger than that noted in Dayal's study of HPC systems at rest [6], though Dayal notes that there is significant variance in average file size from system to system.

IV. Observations

In this section we provide observations and micro-analyses of several facets of the NCAR MSS workload. We take a top-down approach beginning with aggregate observations of the system as a whole, then proceed to looking at user-session behaviors, followed by examining activities on a per-file basis. In Section V, we discuss our observations and provide higher level conclusions and suggestions for future archive designers, as well as suggestions for improving future studies.

A. Aggregate Observations

Observation: Automated file migrations make up the majority of activities.

In our first set of observations, we examine the fraction of actions devoted to file migration. This is of interest as

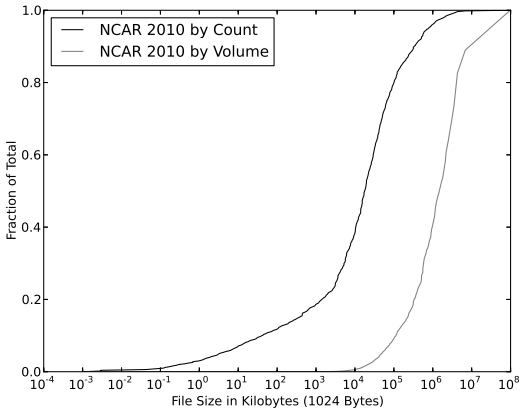


Fig. 3: CDF of file sizes by the fraction of files of a given size, and the file sizes responsible for a given fraction of space.

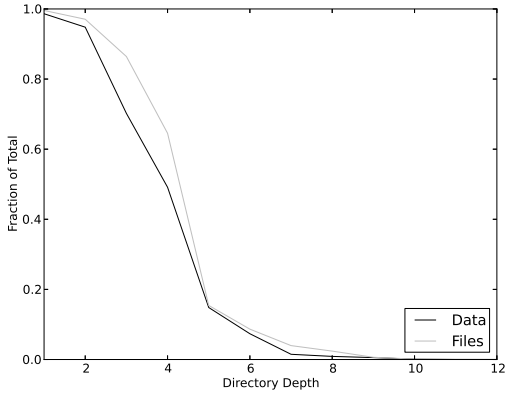
"maintenance" and other supporting actions are easy candidates for optimization—they are predictable and often latency-insensitive.

In Table II, we show the number of user activities and migrate activities by count. Activities due to migration significantly outnumber those from user-sourced reads, creates and writes. This observation is consistent with conclusions drawn from both other archival and enterprise systems in which the dominant fraction of activities are automated "supporting" operations such as integrity checking and metadata manipulations [1, 10, 14]. The reason for this large number of automated actions in the MSS is the use of the disk cache as a staging area file for creations and caching file reads. If the data is not already on tape, it must be copied onto a tape prior to removal from the cache.

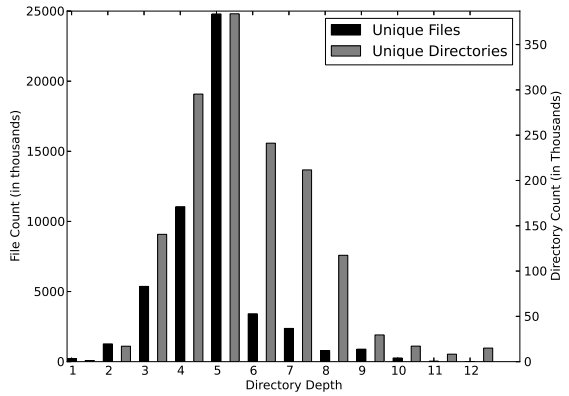
Over the period covered by our sketch, the archive was migrated to a new generation of equipment, ultimately resulting in the entire corpus being both read and written. However, our logs did not note activities from this data migration. This provides us with two key insights. First, the oft-quoted "Write-Once, Read-Maybe" assumption within archives is false from a system maintenance perspective. When we account for these inevitable technology driven data migrations, data is inevitably both read and written. There is still a grain of truth to "Write-Once, Read-Maybe" from the user perspective, but as we discuss later, this assumption is not unequivocally true there either. Second, the lack of evidence of these migrations in our logs highlights the importance of communicating with system administrators and architects to understand the limitations of any data being provided. Without their input we would have been ignorant of the migrations from one set of hardware to the next.

Observation: Most files and data are concentrated around the same relative directory depth, though there is wide variation in the number of files and data in any given directory.

We next examine the distribution of files and data across the namespace because it can offer hints as to how a system should physically group and organize its data. Figure 4 shows



(a) Fraction of files and data below each directory depth. The counts at each depth contain the cumulative sum of all data below the given depth. We truncate the chart at depth 12 since only a very small fraction of files reside at greater depth. Root (depth 0) would automatically contain 100% of data and files.



(b) Number of unique files and directories identified at a given depth. Note the separate axes for directory and file counts; the axes are scaled so that the columns for unique files and directories at depth 5 are the same height.

Fig. 4: Two views of the distribution of files and bytes by directory depth. For example, `/user/foo/` would be at depth 2. The distribution is across 50.5 million files in 1.4 million directories.

two views of the distribution of files at different depths of the namespace.

Figure 4(a) shows a breakdown of the number of files and bytes contained recursively at each directory as a fraction of all the data and files observed. For example, the root of the directory tree, depth 0, would contain 100% of the files and data since it includes everything beneath it. We include files that were purged when calculating the distribution of files and bytes because many files were only observed upon deletion.

In contrast, Figure 4(b) shows the amount of files and data at a *particular* level; approximately 90% of data and individual files are contained at depths 3 through 5, inclusive. Although most files are at depths 3 through 5, there is significant variation in the typical number of files per directory. The median at depth 5 is 11 files per directory, but the mean is 70 and the standard deviation is 607. Since the vast majority

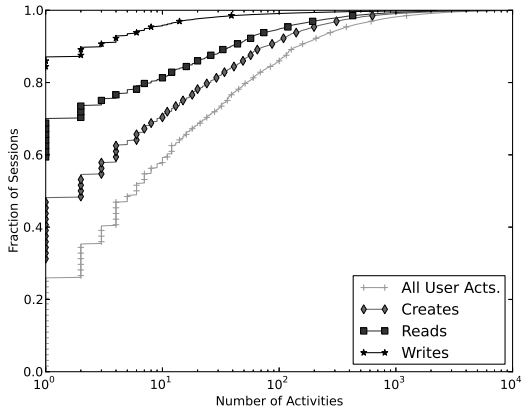


Fig. 5: Breakdown of user activities per session.

of directories contain a modest number of files and bytes physically grouping whole directories on individual physical media is viable, and as we show in our analysis of user behaviors, may yield performance and efficiency benefits.

B. User Behaviors

The raw data we obtained from NCAR did not have user actions grouped into sessions, so to approximate them we artificially grouped activities from individual users into temporally-based sessions. To do this, we used a sliding window of 15 minutes. Any actions (exclusive of purges since they were only logged once per week) that were within 15 minutes of the previous activity for a user were grouped into a session. Any actions that occurred after a 15 minute idle period were put into a new session. The 15 minute idle period was chosen by examining the number of sessions created as the window length grew. Selecting too small of an idle period resulted in many single action sessions, while using an idle period longer than 15 minutes yielded sessions with very few additional actions. Using this method we identified approximately 640,000 unique sessions.

Observation: Most actions come from a relatively small subset of users and sessions. Further, sessions are comprised of only one type of action.

We examine the distribution of activities on a per-user and per-session basis as it allows us to understand typical user-behavior, which in turn is often the behavior for which a system should be optimized.

We find that a relatively small fraction of users are responsible for most actions in the archive, particularly in regards to writes. 20% of the users were responsible for nearly 90% of the logged actions and 90% of the data volume accessed. At the session level, we see a similar distribution of data volume and activities. 10% of the sessions are responsible for nearly 90% of the logged activities and 90% of the observed data movement.

In Figure 5 we show a breakdown of the number of user activities occurring during sessions; the majority of sessions have fewer than 100 actions. Figure 6 shows the sum of file sizes seen in a given session, providing an upper bound on

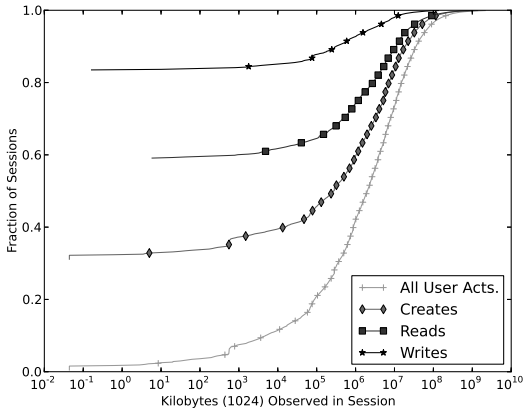


Fig. 6: CDF illustrating the amount of data on which sessions act, broken down by user activity type.

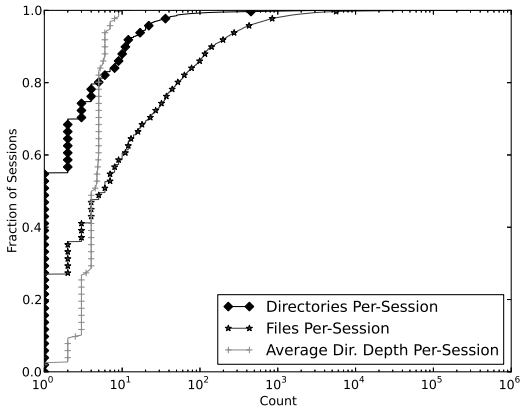


Fig. 7: CDF showing the number of files, directories, and directory depth that sessions act at.

the amount of data that could be manipulated or transferred during the session. Most sessions (over 90%) act on less than 100 GB of data.

We find that sessions never mix user action types. Rather, a given session is comprised of just creates, just reads, or just writes. This makes sense on an intuitive level as it takes time after reading a file to analyze the data and then write results. Moreover, the archive is not a "scratch" space meant for interactive jobs where mixtures of reads and writes are common.

The primary implication we draw here is that there is likely a benefit to having a priority-driven, asynchronous batch interface for very large accesses. Presumably, larger accesses are less latency-sensitive than smaller accesses, providing greater flexibility than hard-coded policies, and allowing large accesses to be intelligently ordered and arranged around smaller, potentially latency-sensitive accesses. Working around latency-sensitive accesses is especially important for systems that may have limited concurrency or drive spin-up policies.

Observation: Most sessions stay within relatively few directories.

We next look at how session actions are distributed through the namespace hierarchy. This can provide hints as to how archival systems should physically group data to reduce seek times and media activations.

In Figure 7 we show the typical directory depth, number of directories, and number of files touched during sessions. The number of directories accessed is typically nearly an order of magnitude less than the number of files, meaning that sessions access multiple files within each directory. The average directory depth per-session is typically (80% of sessions) less than 5, and the directory depth average almost always remains a whole number, suggesting that users tend to access files grouped *at the same depth*. Taken together with earlier observations, this finding suggests that it may be useful to physically group data based on user and directory depth. This technique could yield improved performance, minimizing seeks, and providing for easy pre-fetching and streaming reads and writes for tape and low-power disk-based systems. Physically grouping data by user was similarly done by the RASH portion of NASA's MSS-II in the late 1980s [15, 16] where data was physically grouped by user. Modern systems, such as HPSS [17], also provide tools for the automatic grouping of files to be managed as a single administrative unit. For tape based systems as well as archives built from spun-down disks [18, 19], this can be a boon because it would reduce the need for multiple media activations and mounts.

Observation: Many users did not have any operations beyond deletes associated with them.

We examine the total activity of users to see how many were effectively idle during the trace period. Depending on how the idle users' data is utilized, it may provide a heuristic for purging data.

We identified 1400 users that had user and migration activities associated with them—this is what we focus our session level analyses on. However, we found an additional 200 unique users that were only associated with deleted files. That is, the only activities associated with those users were purge actions. While we do not examine the temporal distribution of individual user activities, this suggests some users and their data—see observations on file lifetimes—may be transient members of the system.

C. File Behaviors

In this section we go deeper and look at the behavior of individual files within the system.

Observation: Around 15% of files are deleted within a year of creation.

We examine file lifetimes for two reasons. First, archives are often considered to be immutable datastores, and as we show, this is not the case. Second, identifying how often, and when, files are deleted can help guide the organization and policies of a system.

In our analysis of file lifetimes, we only consider files for which we observed a create during the logged period. Of the 50 million observed files, we saw 36 million unique file creations, and 11 million deletes. We then correlated approximately 5 million deletes to files with observed creates.

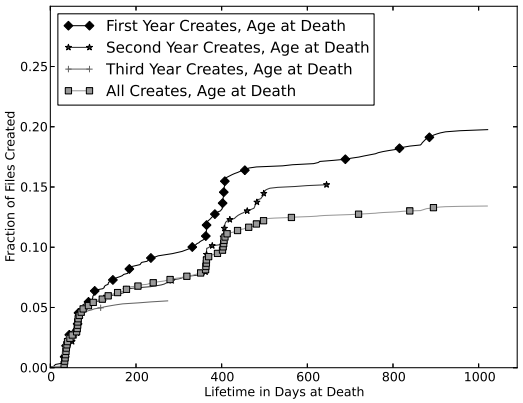


Fig. 8: CDF of the distribution of the lifetime of files created during the trace. "*nth* year creates" refers only to files created during the *nth* year of the trace, while "all creates" refers to *all* files created during the trace period. For example, the "all creates" line ends at around 13.5%, meaning 86.5% of files that were created had not been deleted by the end of the trace.

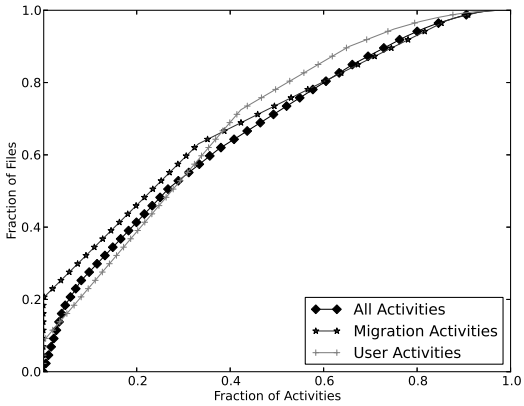


Fig. 9: CDF showing the fraction of files responsible for a given fraction of total activities. Note the user and migration plots do not start at 0 files since not all files had only user activities or only migration activities.

We focus our analysis on these files with observed creates as we can precisely determine their lifetimes, illustrated in Figure 8.

We make two primary observations in this area. First, it appears that approximately 80% of the files created in the first year are in existence at the end of trace, suggesting that the majority of the files have long lifetimes. Second, of those files that *are* deleted, they are most likely to be deleted from the system within one year of creation. While this strong temporal spike is due in part to file retention policies—360 days is one of the most common—it actually makes for a stronger basis in estimating user intentions as they are forced to explicitly decide which files will remain in the system. This is further reinforced by the fact that users are charged for data stored in the archive.

These file lifetime results are interesting for several reasons. First, archives are clearly not immutable data stores—a significant number of files are deleted. Second, while the notion of files being deleted relatively quickly, or not at all, is not new [9, 20, 21], the deletes on the archive occur after months, as opposed to within seconds of creation on enterprise and personal storage. Because of these behaviors it may be useful to add another logical "probationary" level to the storage hierarchy: a place where files that are likely to be deleted are stored before entering a more "permanent" state in the system.

Observation: most files receive few actions. Exclusive of migration activities, 65% are only acted upon once.

We examine the distribution of actions upon files because it can strongly impact caching policies. When examining file activities, we do not include files that are only acted on by a purge. Thus, we study around 43 million files.

Our observation is that, as a whole, actions are evenly distributed across files, and that most files are the target of relatively few actions, as shown in Figure 9, which illustrates the distribution of actions across files. The line is relatively

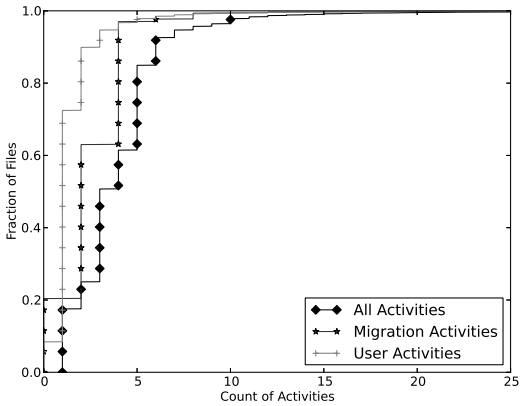


Fig. 10: CDF of the fraction of files receiving particular activities from users (reads,writes, creates) and migration processes. We truncate the count at 25 because only a very small fraction of files receive hundreds or thousands of activities.

flat, indicating that actions are evenly distributed across files; a vanishingly small fraction (less than 0.1%) receive hundreds or thousands of activities. This is further illustrated in Figure 10, where we show the distribution of files by activity count. Across all user activities, 70% of the files we observed only received a single action, usually the initial create of the file.

Our suggestion, based on this observation, is one that NCAR already implements: use the disk cache primarily to absorb creates/writes. Individual file activities are sufficiently spread out and rare that read caching in general would be largely ineffective. The small number of files that are very active, however, could easily be serviced on a disk cache.

Observation: Approximately 5% of files had updates to their data with most updates occurring within one day of another. A small subset of files receive updates over long periods of time (longer than 100 days).

We examine the mutability of files since a common assumption in archival design is the notion of "Write-Once" files.

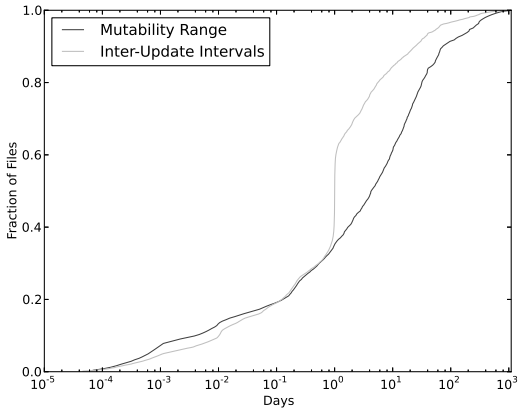


Fig. 11: CDF showing the inter-update interval of files that receive more than a single mutation (more than a single create OR single write), as well as the time range observed between the first and last mutation of a file. These files account for approximately 2.5 million of the observed 50 million files. The x-axis is on a log scale.

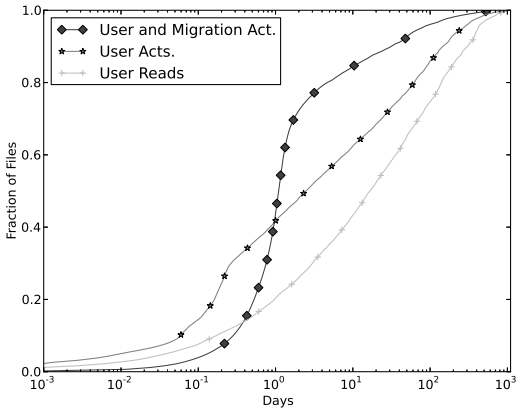


Fig. 12: CDF of the inter-reference interval files under several different filters. "user and migration act'' includes all user and migration activities, while "user acts." includes reads, writes and creates, and "user reads" includes just reads. Note that 100% only accounts for files touched by the respective activities, and files only referenced once are not counted because we are unable to calculate an interval for them.

This can influence caching policies and how data is organized within the system.

Before we move on to our observation, we describe what we consider to be an update, or *mutation* to a file. If a particular file has been created multiple times, all creates after the initial are counted as mutation to file state. A second create to a file overwrites the original data; it may be identical, or entirely different. Any write to a file is treated as an update to its data.

In Figure 11, we show the interval of time between successive updates to the same file. A file is not accounted for if it does not have at least two creates and or writes. Thus, we observe inter-references for 2.5 million files, around 5% of the unique files observed during the trace. Most updates (roughly 65%) occur within one day of another update. After this, we see a marked increase in the interval of time between updates. The next 20% occur within ten days of each other, and the subsequent remainder occur between 10 and 1000 days.

Figure 11 also shows the range of time between a file's initial creation (or first observed write) and its last update or overwrite, which we refer to as the *mutability range*. 40% of the files for which we calculate a mutability range have ranges greater than ten days.

To further explore a file's mutability we count the number of potential updates a file could receive, only counting files that have at least one observed update. This could be either a create action overwriting an existing file or a write action updating a file's data. Over 75% received a single update, and over 95% of files receive fewer than ten updates. The remaining files receive anywhere from 10–100 updates or more. Given the relatively short inter-update time for file data, caching should be able to absorb most updates before writing back to the archive.

Observation: When we account for automated file migrations in a file's inter-reference interval, file inter-reference intervals appear superficially shorter. Considering only user actions causes files to exhibit longer inter-reference intervals, albeit with fewer files accounted for.

We examine inter-reference intervals for two reasons: to explore the impact that file migration has on a file's inter-reference period, and to measure the frequency of accesses to a file. Both factors may impact caching and data placement policies. When calculating the inter-reference interval including migration activities, we treated each migration (read and write, possibly to multiple copies of the same logical file) as a single reference, preventing a misleadingly high inter-reference count for files, particularly those with multiple copies.

Figure 12 shows the inter-reference intervals for files under a variety of filters. Note that in order to be counted a file must have had at least two actions under the relevant filter. When we include migration activities, the general inter-reference interval appears superficially shorter, due to daily migrations of files off of the disk cache. However, when migration activities are filtered out, we see a larger fraction of files under consideration with longer intervals, though this includes *fewer* total files. In essence, automated migration processes can skew how the workload is perceived. Consider, for example, how much the full migration due to the tape library upgrade would have skewed the overall inter-reference period.

The second observation we make is that when we are only concerned with reads, the files that do see repeat reads (approximately 2.2 million) can see very extended periods of time between them. This lines up with the anecdotes we were given about a typical use-case: retrieving stored data at long intervals to validate old experiments and seed new ones.

V. Discussion

While the observations in Section IV are useful on their own, they can provide much-needed implications and guidelines for archive designers when coalesced. Our experiences in analyzing the NCAR MSS traces also lead to suggestions

for long-term storage system designers and analysts that may avoid pitfalls we encountered in our own analysis.

A. Implications Summary and Discussions

Bring back the batch interface. We found that most accesses occur from a subset of users and sessions, and are often very predictable, *e.g.* data migration processes and large user-sourced scripted jobs. Because of these behaviors, there could be significant benefit gained from an asynchronous batch interface that allows a scheduler to intelligently group and place writes while scheduling around more latency sensitive processes. Because end-users are often less tolerant of latency than something like an integrity checking process, the integrity checking can easily be given a lower-priority and run in batch mode in the background, improving user quality-of-service while still providing for administrative and maintenance tasks.

User and namespace grouping may aid archives with offline media. We believe that physically grouping data based on user and namespace heuristics may prove fruitful. We noticed that files and bytes tended to be concentrated around the same levels and that user sessions tended to act within the same directory level. In addition, few files were shared across many users, so grouping by user, like that done in the MSS-II System [15, 16] may be a simple but effective approach to physically grouping data on media, and modern tertiary stores such as HPSS provide tools to largely automate such grouping [17]. This grouping is of prime importance for archives with offline media, such as spun-down disks or unmounted tapes, since they incur high seek penalties and startup costs. This suggestion ties in with the previous suggestion for a batch interface: by delaying groups of writes and batching them, we can better group them based on their user and relationship to the directory hierarchy. If rich metadata are available, more intelligent file grouping and placement techniques can yield significant improvements in access times as demonstrated by Chen *et al.* [22] and their work on multi-dimensional grid data on tertiary storage systems.

Write-Once, Read-Maybe doesn't always hold. An interesting conclusion we have come to is that from a user perspective, the old assumption of "Write-Once, Read-Maybe" is not unequivocally true. As far as "Write-Once" is concerned, while files were generally immutable, a non-trivial fraction were eventually deleted from the archive. A smaller, but also non-trivial fraction received one or more updates to their data. These updates came either through explicit updates to the data, or complete overwrites.

The assumption of "Read-Maybe" appears to at least superficially hold. However, while three years is an enormous length of time compared to most prior studies, we have only seen a fraction of the potential behaviors if the data is intended to survive in perpetuity. We hypothesize that continual data growth rates may superficially mask the real fraction of an archive likely to be read. Consider that we know via communication with administrators that data is often revisited up to *five* years later, yet our logs only covered *three* years of activity. However, when we consider the silent migration of data to new technologies, "Read-Maybe" conclusively becomes "Read-Eventually". As long as the file otherwise survives, it will inevitably be read as it moves to new media.

All told, it is dangerous to rely on any rigid assumptions about the expected read behavior or mutability of files in a system, especially in light of potentially unpredictable impacts of hard policies on user behavior. For example, Holloway found users often attempted to subvert file retention and migration policies through the use of scripts to update file metadata [23].

B. Lessons in Logging and Tracing

In this work, as well as in our prior studies, we have run into many challenges in interpreting and understanding the data we have obtained. We offer suggestions to aid in future tracing and analysis.

Communication is key. If there is a single lesson we have learned in our experiences, it is the importance of communication with system administrators and architects. Time and again we relied on them to understand the system architecture and artifacts of the workload as well as clarify what we did and did not observe in the logs. For example, without their input we would have been ignorant of file migration due to hardware upgrades. Without their shared knowledge we would have had a significantly degraded and even potentially even inaccurate understanding of the archive.

A system start state is invaluable. A good "complete" analysis of a system depends on having *both* snapshots and an activity trace; just one or the other is often insufficient to answer many questions. For example, in this study we lacked a view of the start state of the system that a snapshot could have provided. Because of this we were unable to answer questions such as what fraction of files were left entirely untouched or what fraction of the namespace does a given user occupy. However, having access to only snapshots without a dynamic trace can be equally difficult because it limits the ability to understand the source, timespan, and magnitude of activities occurring in a system.

Don't add semantic meaning to field values. One specific issue we ran into with the NCAR sketch was understanding semantic meanings encoded in field values. For example, we observed situations where the base directory of a path was subtly and silently renamed, *e.g.* `/USER/Foo/bar` to `/uSER/Foo/Bar/` to denote file being in the "trash". This caused our unique file counts to be off by 10% and our directory counts to be off by nearly 20%. Luckily we were able to communicate with an administrator to identify and understand the nature of these silent renames, but this cannot be relied upon for all datasets. As such, we stress to any group that may provide data to others to avoid using field values to encode non-intuitive information unless they are carefully documented.

Be consistent, or be visible. We encountered several subtle format changes in logged activities, including changes as trivial as a field moving one character over or swapping field locations. At best, they cause annoying parser errors and force

data reprocessing with a band-aid fix. At worst, they can silently corrupt results and be extremely difficult to detect. Our suggestion is to ensure that format changes will break a parser, provide an external method for keeping the logs in a consistent format, or document the changes.

Tag or explain activity drop-offs. Sudden reductions in activity rates are difficult to deal with in an analysis since it is often difficult, if not impossible, to determine the true cause of the reduction, *e.g.* it could be due to an actual reduction in the number of activities, a crash, or even a holiday. Without understanding what caused a dip in activity, we have to treat it as null data and discard it, lest we misconstrue what is occurring. Our proposed approach is to have the logger take a more proactive approach and explicitly note when the process that generates the log entries fails or otherwise times out. Another simple fix is to have the logger note any time it starts in the logs it keeps. This can help identify times when the logger is simply inactive, versus ones that are a legitimate reduction in activity rates.

Identify the coverage of a dataset. A question we are always asking about any trace or log we have obtained from outside sources is "what *aren't* we seeing in this trace?". Put another way, we don't know what the *coverage* of a dataset is. For example, we only knew about the data migration from one set of hardware to another via out-of-band communications with system administrators. Our proposed solution is to take a snapshot of a system's state immediately prior to the start of logging/tracing, then after completion of the trace take an additional snapshot. At this point, use the first snapshot and the trace as a delta to create a third *expected* snapshot. We can then compare this expected snapshot with the one taken after the trace and identify where they differ, and help verify what the trace is and is not covering.

VI. Future Work

While we have found several useful results within this study, it is important to note that in this work we are only examining a single relatively specialized archival system. We provide the first such study in nearly 20 years of scientific archive development and use. While we have worked to keep our results and implications as general as possible, further large-scale behavior studies, preferably on different scientific archival storage systems, are necessary. Additional studies will help identify the common and divergent characteristics of scientific archival storage to ensure that results from a single system are not over-generalized to the field as a whole.

Within the NCAR trace, as with any workload study, there are always more tests to run. We intend to further examine the kind of locality users show when accessing files to find more heuristics for physically grouping data. For example, we plan to determine whether data written by a user during a single session is likely to be later accessed as a group. We are also interested in further examining how the active lifetime of a user influences the activity rates of the files they have created. We also have interest in doing a micro-analysis of the raw sensor data NCAR maintains as it may exhibit different usage patterns, however this may prove difficult as it is not always possible to distinctly identify these files.

Based on our experiences in this work as well as other long-term workload analyses, we are in the planning phases of a prototype set of logging framework. The intent of this framework is to aid in the gathering of long-term storage system data with minimal user input, and improve the quality of the logged data. Our proposed framework will use a combination of of metadata snapshots and dynamic storage system traces to identify the coverage of a trace and help users and programmers tune the granularity of long-term activity traces.

Finally, we plan to release anonymized versions of these traces upon publication of our analysis; we have already received permission to do so.

VII. Conclusion

We have conducted the first extensive analysis in nearly 20 years of the long-term behavior of an archival storage system at an HPC data center. Over that 20 year period, archives have grown to encompass tens of petabytes of data across tens of millions of files.

Our analysis found that most users and their associated sessions tend to generate relatively modest amounts of activity, though most actions overall come from large sessions. We also noted that most user sessions stay within a relatively small number of directories at the same depth. Based on these observations, we propose the use of a priority-driven batch interface that physically groups data using heuristics such as by user or by directory depth/subtree to improve data locality. Such a batch interface would improve physical data locality, reducing the number of needed media seeks and mounts, while also allowing large latency insensitive operations, such as file migration tasks, to be scheduled around latency-sensitive accesses. We also found that the notion of "Write-Once, Read-Maybe" scientific data archival storage is weakening: non-trivial numbers of files show mutability and/or deletions. Further, when we include technology migrations, files are no longer "read-maybe" but rather "read-eventually". We also found that, in contrast to many other workloads, file usage is evenly spread, with few files appreciably more popular than others, limiting the effectiveness of read caching.

By better understanding the usage and behavior of a heavily-used scientific data archive in an HPC data center, we have provided valuable suggestions to improve both design and management of future large-scale scientific archives.

Acknowledgments

This work was supported in part by the NSF under awards CNS-0917396 (part of the American Recovery and Reinvestment Act of 2009 [Public Law 111-5]), IIP-0934401 and CCF-0937938, the Department of Energy under Award Number DE-FC02-10ER26017/DE-SC0005417, and the industrial members of the Storage Systems Research Center (SSRC) and the Center for Research in Intelligent Storage (CRIS). Additional thanks to our colleagues in the SSRC for their valuable help and feedback.

REFERENCES

[1] I. F. Adams, E. L. Miller, and M. W. Storer, "Analysis of workload behavior in scientific and historical long-term data repositories," *ACM Transactions on Storage*, vol. 8, no. 2, 2012.

[2] J. C. Frank, E. L. Miller, I. F. Adams, and D. C. Rosenthal, "Evolutionary trends in a supercomputing tertiary storage environment," in *Proceedings of the 20th IEEE International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2012.

[3] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch, "A five-year study of file-system metadata," in *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST)*, Feb. 2007, pp. 31–45.

[4] T. Gibson, E. L. Miller, and D. D. E. Long, "Long-term file activity and inter-reference patterns," in *Proceedings of the 24th International Conference for the Resource Management and Performance and Performance Evaluation of Enterprise Computing Systems (CMG98)*. Anaheim, CA: CMG, Dec. 1998, pp. 976–987.

[5] H. Cho, S. Kim, and S. Lee, "Analysis of long-term file system activities on cluster systems," *World Academy of Science, Engineering and Technology*, vol. 60, 2009.

[6] S. Dayal, "Characterizing HEC Storage Systems at Rest," Carnegie Mellon University, Tech. Rep. CMU-PDL-08-109, 2008.

[7] F. Wang, Q. Xin, B. Hong, S. A. Brandt, E. L. Miller, D. D. E. Long, and T. T. McLarty, "File system workload analysis for large scale scientific computing applications," in *Proceedings of the 21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies*, College Park, MD, Apr. 2004, pp. 139–152.

[8] E. Anderson, "Capture, conversion, and analysis of an intense NFS workload," in *Proceedings 7th USENIX Conference on File and Storage Technologies (FAST'09)*, 2009, pp. 139–152.

[9] A. W. Leung, S. Pasupathy, G. Goodson, and E. L. Miller, "Measurement and analysis of large-scale network file system workloads," in *Proceedings of the 2008 USENIX Annual Technical Conference*, Jun. 2008.

[10] Y. Chen, K. Srinivasan, G. Goodson, and R. Katz, "Design implications for enterprise storage systems via multi-dimensional trace analysis," in *Proceedings of the 23rd Symposium on Operating Systems Principles (SOSP'11)*, Cascais, Portugal, 2011, pp. 43–56.

[11] A. J. Smith, "Analysis of long term file reference patterns for application to file migration algorithms," *IEEE Transactions on Software Engineering*, vol. 7, no. 4, pp. 403–417, Jul. 1981.

[12] E. Miller and R. Katz, "An analysis of file migration in a Unix supercomputing environment," in *Proceedings of the Winter 1993 USENIX Technical Conference*, Jan. 1993, pp. 421–433. [Online]. Available: http://www.ssrc.ucsc.edu/~elm/Papers/usenix93.pdf

[13] D. W. Jensen and D. A. Reed, "File archive activity in a supercomputing environment," in *Proceedings of the International Conference on Supercomputing (ICS 1993)*, Tokyo, Japan, Jul. 1993, pp. 387–396.

[14] W. Vogels, "File system usage in Windows NT 4.0," in *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP '99)*, Dec. 1999, pp. 93–109.

[15] R. L. Henderson and A. Poston, "MSS-II and RASH: A mainframe UNIX based mass storage system with a rapid access storage hierarchy file management system," in *Proceedings of the Winter 1989 USENIX Technical Conference*, 1989, pp. 65–84.

[16] D. Twenten, "Hiding Mass Storage Under Unix: NASA's MSS-II Architecture," in *Proceedings of Mass Storage Systems, Crisis in Mass Storage*, 1990.

[17] "HPSS installation guide, release 7.3 (revision 1)," www.hpss-collaboration.org/documents/hpss731/install_guide.pdf, May 2010.

[18] M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti, "Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST)*, Feb. 2008.

[19] D. Colarelli and D. Grunwald, "Massive arrays of idle disks for storage archives," in *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing (SC '02)*, Nov. 2002.

[20] M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. Shirriff, and J. K. Ousterhout, "Measurements of a distributed file system," in *Proceedings of the 13th ACM Symposium on Operating Systems Principles (SOSP '91)*, Oct. 1991, pp. 198–212.

[21] D. Roselli, J. Lorch, and T. Anderson, "A comparison of file system workloads," in *Proceedings of the 2000 USENIX Annual Technical Conference*. San Diego, CA: USENIX Association, Jun. 2000, pp. 41–54.

[22] L. Chen, R. Drach, M. Keating, S. Louis, D. Rotem, and A. Shoshani, "Efficient organization and access of multi-dimensional datasets on tertiary storage systems," in *Information Systems*. Elsevier, 1995, vol. 20, no. 2, pp. 155–183.

[23] A. Holloway, "The purge threat: Scientists' thoughts on usability in peta-scale," in *Proceedings of the 6th Petascale Data Storage Workshop (PDSW '11)*, November 2011.